IBM Developer
SKILLS NETWORK

Webscraping in R

Estimated time needed: **15** minutes

Objectives

After completing this lab you will be able to:

- Understand HTML via coding practice
- Perform basic webscraping using rvest

Table of Contents

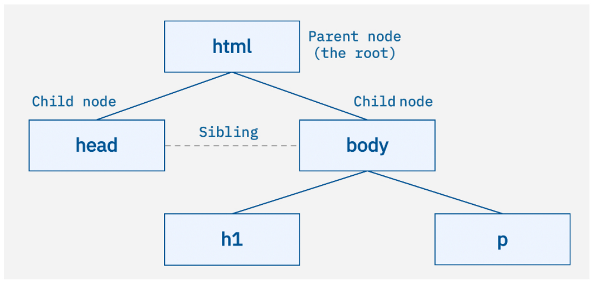
Overview of HTML

- HTML stands for Hypertext Markup Language and it is used mainly for writing web pages.
- An HTML page consists of many organized HTML nodes or elements that tell a browser how to render its content.
- Each node or element has a start tag and an end tag with the same name and wraps some textual content.

One key feature of HTML is that nodes can be nested within other nodes, organizing into a tree-like structure like the folders in a file system. Below is a basic HTML node structure:

- `<html>` node is the root node,
- `<html>` node has two children: `<head>` and `<body>`.
- Since the `<head>` and `<body>` nodes have the same parent `<html>` node they are siblings to each other.
- Similarly, the `<body>` node has two child nodes, the `<h1>` and `<p>` nodes.

It is important to understand this tree-structure when writing a new HTML page or extracting data from an existing HTML page.



The `revest` library

The `rvest` package is a popular web scraping package for R. After rvest reads an HTML page, you can use the tag names to find the child nodes of the current node.

```
[ ]: library(rvest)
```

We also need to import httr library to get some HTML pages by sending HTTP GET request

```
[ ]: library(httr)
```

First let's warm-up by reading HTML from the following character variable `simple_html_text`

```
[ ]: # A simple HTML document in a character variable
     simple_html_text <- "
     <html>
         <body>
             <p>This is a test html page</p>
         </body>
     </html>"
```

Then use the read_html function to create the HTML root node, i.e., the html node by loading the `simple_html_text`

```
[ ]: # Create the root html node by reading the simple HTML string
     root_node <- read_html(simple_html_text)
     root_node
```

You can also check the type of `root_node`

```
[ ]: class(root_node)
```

You can see the class is `xml_node` because rvest load HTML pages and convert them using XML format internally. XML has similar parent-child tree structure but more suitable for storing and tranporting data than HTML.

Next, let's try to create a HTML node by loading a remote HTML page given a URL

```
[ ]: ibm_html_node <- read_html("http://www.ibm.com")
     ibm_html_node
```

Sometimes you want to download some HTML pages and analyze them offline, you could use `download.file` to do so:

```
[ ]: # download the R home page and save it to an HTML file locally called r.html
     download.file('https://www.r-project.org', destfile = 'r.html')
```

```
[ ]: # Create a html_node from the local r.html file
     html_node <- read_html('r.html')
     html_node
```

Coding Exercise: in the code cell below, download a html node using any URL you like.

```
[ ]: # Write your code below. Don't forget to press Shift+Enter to execute the cell
```

Now you know how to read an HTML page from a character variable, a URL, or a local HTML file. Next let's see how to parse and extract data from a specific node(s) starting from the root node

```
[ ]: simple_html_text <- "
     <html>
         <body>
             <p style=\"color:red;\">This is a test html page</p>
         </body>
     </html>"

     root_node <- read_html(simple_html_text)
     root_node
```

Get the `<body>` node by using its parent node `<html>`

```
[ ]: # Get the child <body> node from <html> root_node
     body_node <- html_node(root_node, "body")
     body_node
```

You can see it has a child node paragraph `<p>`

Let's get the content of the `<p>`

```
[ ]: # Get the child <p> node from its <body> node
     p_node <- html_node(body_node, "p")
     p_content<-html_text(p_node)
     p_content
```

The `<p>` node also has `style` attribute with value `color:red;`, which means we want the browser to render its text using red color. To get an attribute of a node, we can use a function called `html_attr("attribute name")`

```
[ ]: # Use the p_node as the first argument to get its attribute
     style_attr <- html_attr(p_node, "style")
     style_attr
```

In the code cell below, the downloaded `r.html` file (from `https://www.r-project.org`) has an `<img>` node representing an image URL to R logo image (a relative path on its web server), let's try to find the image URL and download it.

Your need to paste the relative path in `<img>` with the the `https://www.r-project.org` to get the full URL of the image, and use the `GET` function to request the image as bytes in the response

```
[ ]: # Write your code below. Don't forget to press Shift+Enter to execute the cell
     url <- 'https://www.r-project.org'
     html_node <- read_html('r.html')

     # Get the image node using its root node
     img_node <- html_node(html_node, "img")
     # Get the "src" attribute of img node, representing the image location
     img_relative_path <- html_attr(img_node, "src")
     img_relative_path

     # Paste img_relative_path with 'https://www.r-project.org'
     image_path <- paste(url, img_relative_path, sep="")
     # Use GET request to get the image
     image_response<-GET(image_path)
```

Then use `writeBin()` function to save the returned bytes into an image file.

```
[ ]: # Parse the body from the response as bytes
     image <- content(image_response, "raw")
     # Write the bytes to a png file
     writeBin(image, "r.png")
```

Now, from the file list on the left, you should be able to find a saved r.png file.

In HTML, many tabluar data are stored in `<table>` nodes. Thus, it is very important to be able to extract data from `<table>` nodes and preferably convert them into R data frames.

Below is a sample HTML page contains a color table showing the supported HTML colors, and we want to load it as a R data frame so we can analyze it using data frame-related operations.

```
[ ]: table_url <- "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/datasets/HTMLColorCodes.html"
```

Like other HTML nodes, let's first get the `<table>` node using `html_node` function

```
[ ]: root_node <-read_html(table_url)
     table_node <- html_node(root_node, "table")
     table_node
```

You can see the table node in a messy HTML format. Fortunately, you dont need to parse it by yourself, `rvest` provides a handy function called `html_table()` to convert `<table>` node into R dataframe

[ ]:

```
# Extract content from table_node and convert the data into a dataframe
color_data_frame <- html_table(table_node)
head(color_data_frame)
```

But you could see the table headers were parsed as the first row, no worries, we could manually fix that

[ ]:

```
names(color_data_frame)
```

[ ]:

```
# Convert the first row as column names
names(color_data_frame) <- as.matrix(color_data_frame[1, ])
# Then remove the first row
data_frame <- color_data_frame[-1, ]

head(data_frame)
names(color_data_frame)
```

That's it for `webscraping` in R, there is a lot more you can do check out rvest.

Did you know? IBM Watson Studio lets you build and deploy an AI solution, using the best of open source and IBM software and giving your team a single environment to work in. Learn more here.

Authors

Hi, this is Yan Luo the author of this notebook.

I hope you found it easy to learn how to do webscraping in R! Feel free to connect with us if you have any questions.

Other Contributors

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-03-05	1.0	Yan	Initial version created

Simple 0 1 Initialized (additional servers needed) R | Idle Mem: 229.38 / 6144.00 MB Mode: Command Ln 1, Col 1 English (American) lab3_jupyter_webscraping.ipynb